2018 26th Mediterranean Conference on Control and Automation (MED)
Zadar, Croatia, June 19-22, 2018

Ground Extraction from 3D Lidar Point Clouds with the Classification Learner App*

Antonio Pomares, Jorge L. Martínez[1], Anthony Mandow,
María A. Martínez, Mariano Morán and Jesús Morales

***Abstract*— Ground extraction from three-dimensional (3D) range data is a relevant problem for outdoor navigation of unmanned ground vehicles. Even if this problem has received attention with specific heuristics and segmentation approaches, identification of ground and non-ground points can benefit from state-of-the-art classification methods, such as those included in the Matlab Classification Learner App. This paper proposes a comparative study of the machine learning methods included in this tool in terms of training times as well as in their predictive performance. With this purpose, we have combined three suitable features for ground detection, which has been applied to an urban dataset with several labeled 3D point clouds. Most of the analyzed techniques achieve good classification results, but only a few offer low training and prediction times.**

I. INTRODUCTION

Three-dimensional (3D) point clouds obtained from on-board sensors like time-of-flight cameras, 3D laser scanners, or stereo-vision systems [1], constitute a reliable source of information for mobile robots [2]. Usually, 3D point clouds need to be processed to determine traversable, forbidden or uncertain areas on the terrain [3]. Furthermore, ground extraction can be a fundamental step prior to the segmentation of objects in the scene [4][5].

Many authors have addressed the crucial problem of ground detection with specific segmentation algorithms that use data characteristics such as height information, normals, gradients, and thresholds [6]. Different machine learning techniques have been proposed to classify 3D point clouds in categories such as ground, poles, wires or vegetation [7][8]. With supervised machine learning, it is necessary to provide representative examples previously labeled to obtain useful predictive models that can be applied to new data [9].

Within its latest releases, the Matlab software has incorporated the Statistics and Machine Learning toolbox [10], which contains a considerable number of machine learning techniques. The Classification Learner App included in this toolbox allows a quick access to supervised learning methods. The methods included in this toolbox can be employed for a great variety of real world applications such as object detection in image processing [11] or brain-computer interfaces [12].

This paper proposes a comparative analysis of all the machine learning methods included in the Classification Learner App applied to ground detection from 3D point clouds. With this purpose, we have selected a set of three appropriate features for ground identification. Then, 23 state-of-the-art classification methods have been compared in terms of training times and predictive performance. In particular, the analysis has been based on several labeled 3D point clouds from a representative urban dataset.

The paper is organized as follows. Next section overviews the Classification Learner App of Matlab. Feature extraction from a 3D point cloud for ground segmentation is presented in section III. Then, experimental results are analyzed in section IV. Finally, the conclusions section offers a discussion of lessons learned and an outlook for future work.

II. THE CLASSIFICATION LEARNER APP

This section reviews the Classification Learner App, which is contained in the latest versions of the Statistics and Machine Learning toolbox of Matlab. This application allows automated training for the set of state-of-the-art supervised machine learning classifiers included in the toolbox. Training requires a set of input data and known outputs to this data (i.e., labeled classes). Then, trained classifiers can be exported to the Matlab workspace, where they can be employed to compute predictions for new input data by using the Matlab function `predictFcn`.

With the R2016b version of Matlab, the App supports a total of 23 classifier types, which can be organized in six major classification algorithms:

- *Support vector machine* (SVM), where mathematical functions determine the limits between classes. The available types of SVMs are Linear, Quadratic, Cubic, Fine Gaussian, Medium Gaussian and Coarse Gaussian.
- *Decision tree*, where classes are predicted by choosing branches in a tree starting from the root to the leaf nodes. Types of decision trees include Simple, Medium and Complex.
- *K nearest neighbors* (KNN), where data is classified according to the class of its K nearest neighbors. KNN has several types: Fine, Medium, Coarse, Cosine, Cubic, and Weighted.
- *Discriminant analysis*, which finds combinations of features that characterize or separate classes. Discriminant analysis can be Linear or Quadratic.

*This work was partially supported by the Spanish project DPI 2015-65186-R.

[1]Corresponding author. Email: jlmartinez@uma.es, Phone: +34-951-952- 322

All authors are with the Universidad de Málaga, Andalucía Tech, Dpto. Ingeniería de Sistemas y Automática, Escuela de Ingenierías Industriales, Calle Dr. Ortiz Ramos s/n, 29071-Málaga, Spain Emails: anpol92@hotmail.com, amandow@uma.es, mamartinezs@uma.es, marianomoran@hotmail.com, jesus.morales@uma.es

- *Logistic regression*, that involves a probabilistic approach for binary classification, where a logistic function is fitted to the feature space.
- *Ensemble classification*, where two or more classification methods are combined to improve their individual performance. Different types are available: Bagged Trees, Boosted Trees, Subspace Discriminant, Subspace KNN and RUSBoosted Trees.

Furthermore, the Classification Learner App offers the following built-in validation schemes that indicate the predictive accuracy of the trained model:

- *No validation*. All input data is used for training the model. Then, the App computes the confusion matrix by using the same training data.
- *Holdout validation*. The input data is divided into two complementary sets: one is used for training and the other to validate the resulting model.
- *Cross-validation*. This method selects q disjoint sets to partition the data. While only one set is used for the validation of the model, the other $q-1$ are used for training. This process is repeated q times and the resulting confusion matrix is obtained with the arithmetic means of the results at each iteration. This is the default validation option, with $q = 5$.

Independently of the chosen scheme, the final predictive model is always trained using the full data set.

III. FEATURE COMPUTATION

The definition of a suitable set of features is an important issue for successful 3D point cloud classification [13]. Supervised classification can be performed in an individual basis, where each point is assigned independently of its neighbors' class [14]. Nevertheless, the spatial features used in classification need to be extracted from the neighborhood of each 3D point, which can be computed by using a fixed radius of proximity [15].

The commonly used approach for nearest neighbor search is based on a k-d-tree data structure due to its computational efficiency [16]. To this end, the Matlab function `rangesearch` has been employed with $k = 3$ and a fixed radius of $0.3\,\mathrm{m}$. Those 3D points that have less than five neighbors are discarded from feature calculation (and classification) to ensure a minimum of spatial information that can lead to reliable features.

Spatial distribution can be studied through principal components analysis (PCA) [15]. PCA begins with the $n \times 3$ matrix P_i that contains the Cartesian coordinates in a global frame XYZ of point p_i and all its neighbors. Then, the 3×3 symmetric definite-positive covariance matrix S_i associated to p_i is calculated as:

$$S_i = \frac{(P_i - \overline{P_i})^T (P_i - \overline{P_i})}{n}, \tag{1}$$

where ^{T} represents the transpose operation, and $\overline{P_i}$ is a $n \times 3$ matrix that contains the mean value of P_i: $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ in each of its n rows. The eigenvalues λ associated with S_i are calculated with the characteristic equation:

$$|S_i - \lambda\, I| = 0, \tag{2}$$

where I is a 3×3 identity matrix. Eigenvalues are positive real numbers that can be ordered as:

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3. \tag{3}$$

The eigenvectors v associated with each eigenvalue λ are calculated with the equation:

$$S_i\, v = \lambda\, v. \tag{4}$$

This analysis can be performed with the Matlab functions `cov` and `eig`.

In this work, we combine three features $\{f_1, f_2, f_3\}$ commonly employed for ground identification. For leveled 3D point clouds, the minimum height (f_1) is the most relevant geometric feature for terrain classification [17]. In addition, the vertical orientation (f_2) and the scatterness (f_3) of the local spatial shape can be employed as features to detect traversable ground [18]. This selection keeps a reduced set of features that are relevant to classify ground points.

Thus, the three real numbers used as features to classify every p_i are calculated as:

- *Minimum height*: The minimum Z coordinate among all the points p_j of P_i:

$$f_1(P_i) = \min_{\forall p_j}(z_j). \tag{5}$$

- *Vertical orientation*: The slope can be obtained from an eigenvector v_1 of the lowest eigenvalue λ_1:

$$f_2(P_i) = \arccos((0,0,1) \cdot v_1). \tag{6}$$

- *Scatterness*: The smallest eigenvalue can be employed directly as an spatial dispersion index:

$$f_3(P_i) = \lambda_1. \tag{7}$$

IV. EXPERIMENTAL ANALYSIS

This section proposes a comparative analysis of all the machine learning methods included in the Classification Learner App when applied to ground detection from 3D point clouds with the set of features defined in (5)-(7). All in all, 23 state-of-the-art classification methods have been compared in terms of training times and predictive performance. Experiments have been carried out on a computer with an Intel Core processor i7 at $3.5\,\mathrm{GHz}$ and 16 GB RAM. To make the analysis manageable, the default options for each method have been applied.

In the experimental analysis, accuracy is considered as a performance index, which is computed as:

$$ACC = 100 \cdot \frac{TP + TN}{TP + FP + TN + FN}, \tag{8}$$

where TP, FP, TN, and FN are the number of true positives, false positives, true negatives, and false negatives, respectively. In our binary classification problem, positive refers to "ground" and negative to "non ground."

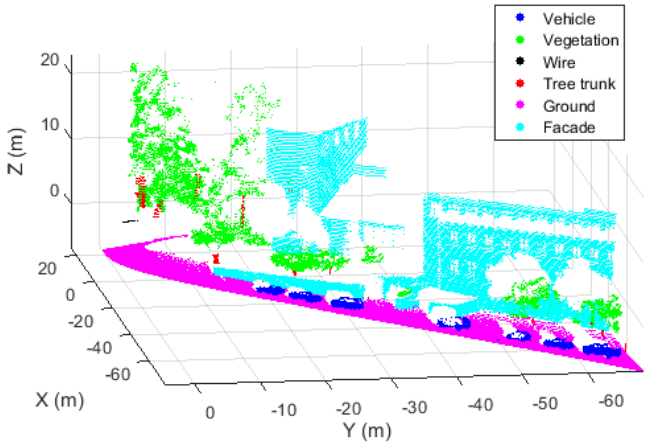

Fig. 1. Example of a classified 3-D laser scan from the `VMR-Oakland-v2` dataset [22].

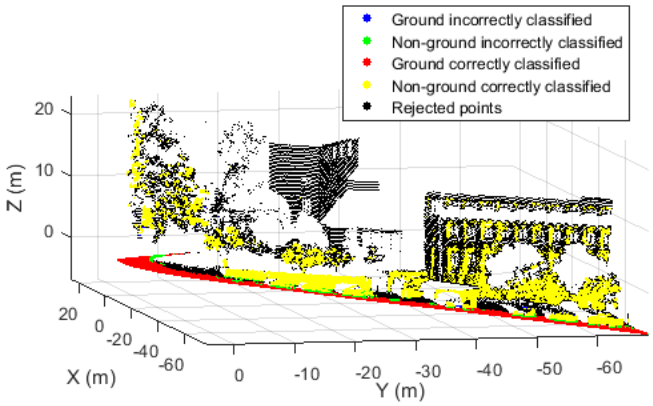

Fig. 2. Classification results with the Medium Gaussian SVM algorithm for the 3D laser scan of Figure 1.

A. Dataset

Some published works on 3D point cloud classifications in outdoor environments have made available their labeled datasets through public repositories [13][19][20][21]. The one chosen for this work is the *VMR-Oakland-v2* dataset that contains leveled urban 3D laser scans acquired at the ground level [22].

Figure 1 shows a labeled 3D laser scan from the *VMR-Oakland-v2* repository, where several classes, including "ground", can be observed. To perform binary classification, we have grouped together the points labeled as "vehicle", "vegetation", "wire", "tree trunk" and "facade" into the "non-ground" class.

From this repository, six labeled 3D scans have been chosen for training and another six for validation. All in all, 561277 points have been used for training and 495770 points for the off-line validation (different from the App's built-in validation). The time employed for extracting features has been 16.12 s and 15.33 s for training and validation data, respectively. Due to the restriction of a minimum number of 5-neighbors to extract reliable features, the number of used points is reduced to 458468 and 421995 for training and off-line validation, respectively.

B. Training

Table I shows the training time (t) and the estimated prediction accuracy (ACC) for every training method when using the *No-Validation* and the *Cross-Validation* (with $q = 5$) options. In this case, the ACC values have been computed by using the confusion matrix provided by the App's built-in validation.

As expected, the *Cross-Validation* option imposes a noticeable overhead to training time when compared to *No-Validation*. *Cross-Validation* does not alter the resulting model, but is intended to give a better estimate of its predictive accuracy. However, in these experiments, the ACC resulting from both built-in validation options yield similar values, with those of *Cross-Validation* being slightly less optimistic.

The estimated accuracy of the training functions obtained with *Cross-Validation* is very high, with a mean value of 95.3 %. The only major exception comes from the Cubic SVM (# 3) with a small percentage of only 22.3 %. This exception represents an outlier with respect to the whole range of classifiers, including the rest of SVM algorithms.

The average training time for the 23 methods with *No-Validation* is very high: 14 000 s, i.e., about four hours. Three SVM methods (#1, #2, #3) require enormous training times of around 35 000 s. On the other hand, the Decision Tree (#7, #8, #9), Discriminant Analysis (#16, #17), and Logistic Regression (#18) methods stand out with low learning times below 25 s. All in all, a dramatic difference of training time can be observed between the fastest methods and the rest of classifiers.

C. Validation

Trained classifiers have been validated with new input data independently of the App's built-in validation. Table II shows the classification time (t), the confusion matrix values, and the resulting accuracy (ACC).

It can be observed that classification times, with a mean value of of 151 s, are considerably shorter than training times. The Cosine KNN method (#13) stands out as the slowest, with 2822 s (i.e., 47 minutes approximately). Conversely, the Medium Tree (#8), Complex Tree (#9) and Quadratic Discriminant (#17) methods are well below 1 s.

The average accuracy (94.0 %) is similar to the one obtained with the built-in *cross-validation* results. As before, the predictive model with the lowest percentage of success is Cubic SVM (#3), but with a higher value of 42.4 %.

Figure 2 illustrates the classification results obtained with Medium Gaussian SVM (#5) applied to the 3D point cloud of Figure 1. Unclassified points due to the neighborhood feature restriction (depicted in black) correspond mostly to distant points. In particular, from 97207 points of this 3D scan, 18295 do not reach the minimum number of neighbors (18.82%). False positives, shown in green, correspond to points very close to the ground. Thus, from 13953 non-ground points, 1119 have been classified incorrectly (8.72%).

TABLE I

TRAINING RESULTS FOR "GROUND" CLASSIFICATION.

#	Training method	*No validation* t (s)	*No validation* ACC (%)	*Cross-validation* t (s)	*Cross-validation* ACC (%)
1	Linear SVM	33553.0	98.9	151800.3	98.9
2	Quadratic SVM	36470.1	94.0	162090.0	91.9
3	Cubic SVM	33895.2	27.0	156110.1	22.3
4	Fine Gaussian SVM	6445.9	99.3	22534.4	99.2
5	Medium Gaussian SVM	8379.3	99.2	29452.8	99.2
6	Coarse Gaussian SVM	9449.1	99.1	33190.6	99.1
7	Simple Tree	7.8	98.8	17.2	98.8
8	Medium Tree	8.4	99.2	22.3	99.2
9	Complex Tree	11.5	99.3	38.7	99.3
10	Fine KNN	9457.9	100.0	33214.0	99.1
11	Medium KNN	9473.3	99.4	33246.2	99.3
12	Coarse KNN	9538.1	99.2	33332.5	99.2
13	Cosine KNN	20110.3	99.2	41620.4	99.1
14	Cubic KNN	20156.0	99.3	41695.0	99.3
15	Weighted KNN	20172.1	100.0	41723.2	99.3
16	Linear Discriminant	9.5	97.0	12.2	97.0
17	Quadratic Discriminant	12.9	98.8	15.3	98.8
18	Logistic Regression	23.3	98.6	58.1	98.6
19	Bagged Trees	20913.0	100.0	44541.6	99.4
20	Boosted Trees	20348.0	99.2	42393.6	99.2
21	Subspace Discriminant	20998.4	96.9	44764.2	97.0
22	Subspace KNN	21231.2	100.0	45310.8	99.0
23	RUSBoosted Trees	21339.0	99.2	45725.9	99.2

TABLE II

VALIDATION RESULTS FOR "GROUND" CLASSIFICATION.

#	Predictive model	t (s)	ACC (%)	TP	FP	TN	FN
1	Linear SVM	83.90	99.1	333508	1649	84702	2136
2	Quadratic SVM	46.60	94.5	334390	767	64324	22514
3	Cubic SVM	13.96	42.4	117811	217346	61068	25770
4	Fine Gaussian SVM	109.50	95.1	315338	19819	85764	1074
5	Medium Gaussian SVM	100.64	99.2	333048	2109	85612	1226
6	Coarse Gaussian SVM	132.10	99.2	333261	1896	85369	1469
7	Simple Tree	1.54	98.9	332638	2519	84692	2146
8	Medium Tree	0.27	99.1	332711	2446	85439	1399
9	Complex Tree	0.26	97.2	324389	10768	85643	1195
10	Fine KNN	2.93	86.8	281047	54110	85227	1611
11	Medium KNN	5.95	92.7	305656	29501	85550	1288
12	Coarse KNN	83.32	99.0	332200	2957	85602	1236
13	Cosine KNN	2822.02	99.1	332718	2439	85596	1242
14	Cubic KNN	21.11	92.5	304856	30301	85548	1290
15	Weighted KNN	6.11	92.2	303409	31748	85608	1230
16	Linear Discriminant	2.76	97.3	334211	946	76525	10313
17	Quadratic Discriminant	0.64	99.0	332425	2732	85275	1563
18	Logistic Regression	1.58	98.9	333914	1243	83392	3446
19	Bagged Trees	11.45	97.0	323678	11479	85727	1111
20	Boosted Trees	7.25	99.1	332302	2855	85805	1033
21	Subspace Discriminant	9.14	97.3	334214	943	76237	10601
22	Subspace KNN	9.00	85.8	277233	57924	84990	1848
23	RUSBoosted Trees	6.10	99.1	332141	3016	85833	1005

As for false negatives (shown in blue), from 64959 ground points, only 155 have been classified incorrectly (0.24%).

V. CONCLUSIONS

In this paper, we have addressed identification of ground and non-ground points from 3D lidar data, which is a relevant problem for outdoor navigation of unmanned ground vehicles. In particular, we have proposed a comparative study of 23 state-of-the-art machine learning methods that are available through the Matlab Classification Learner App. With this purpose, we have combined three features suitable for ground detection, which has been applied to an urban dataset with several labeled 3D point clouds.

In general, the validation results have shown a high accuracy, but some methods have required a very high training time. Methods such as decision trees, discriminant analysis, and logistic regression have shown a very good performance in terms of training and prediction times as well as in prediction accuracy. These three methods are among the simpler supervised learning techniques available on the App. This result suggests that ground extraction from leveled 3D point clouds of urban environments does not require complex

Fig. 3. Photograph of the park scene.

classification algorithms. Thus, classifiers like the Medium Decision Tree (# 8) have offered fast and very accurate results that could be a suitable alternative to heuristic ground segmentation methods in robotic applications.

A. Outlook

The real-time application of the binary classifiers for autonomous navigation of a mobile robot in natural terrains is a future issue of interest. As a matter of example, the predictive models obtained in Section IV have been applied, without further training, to a park scene captured with the UNOlaser 3D rangefinder [23] located $1\,\mathrm{m}$ above the ground (see Fig. 3).

Figure 4 shows excellent, acceptable and unsatisfactory classification results provided by the Weighted KNN (# 15), the Medium Tree (# 8) and the Cubic SVM (# 3) models, respectively. It is remarkable that only $0.26\,\%$ of data (i.e, 345 points) have been discarded from classification (black dots) due to the 5-neighbors restriction in comparison with the $16.71\,\%$ of the *VMR-Oakland-v2* dataset. This can be explained because the maximum range of the 3D sensor is limited to $30\,\mathrm{m}$ in the park scene.

From 130585 3D points, the Weighted KNN model has classified 105688 as ground (red dots) and 24552 as non-ground (yellow dots). The Medium Tree model have classified more points as ground (106071), which includes some false positives on the tree crowns, and less as non-ground (24169). In both cases, the ground points close to the tree trunks are incorrectly classified as non-ground.

In view of these promising results, the performance of the predictive models to classify non-urban ground will be evaluated in a future work. Furthermore, in order to include non-traversable areas, such as steep slopes, into the non-ground class, it will be necessary to prepare appropriate training data sets and to evaluate different feature combinations.

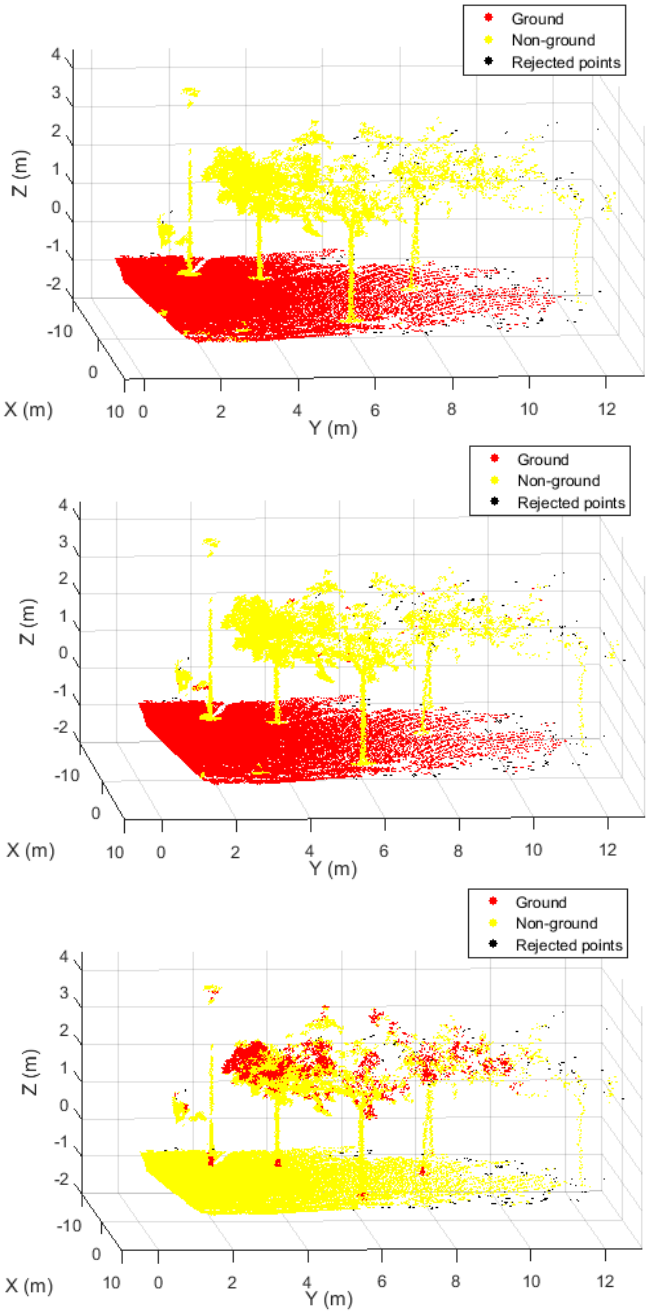

Fig. 4. Classification results for a 3D point cloud from the park with Weighted KNN (top), Medium Tree (middle) and Cubic SVM (bottom).

References

[1] J. Poppinga, A. Birk, and K. Pathak, "A characterization of 3D sensors for response robots," *Lecture Notes in Computer Science*, vol. 5949, pp. 264–275, 2010.

[2] H. Liu, G. He, H. Yu, Y. Zhuang, and W. Wang, "Fast 3D scene segmentation and classification with sequential 2D laser scanning data in urban environments," in *Proc. 35th Chinese Control Conference*, Chengdu, China, 2016, pp. 7131–7136.

[3] C. Castejón, D. Blanco, and L. Moreno, "Compact modeling technique for outdoor navigation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, pp. 9–24, 2008.

[4] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D lidar point clouds," in *Proc. IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2798–2805.

[5] J. Balado, L. Díaz-Vilariño, P. Arias, and H. González-Jorge, "Automatic classification of urban ground elements from mobile laser scanning data," *Automation in Construction*, vol. 86, pp. 226–239, 2018.

[6] H. Vu, H. T. Nguyen, P. M. Chu, W. Zhang, S. Cho, Y. W. Park, and K. Cho, "Adaptive ground segmentation method for real-time mobile robot control," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, 2017.

[7] J. Lalonde, N. Vandepel, D. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility." *Journal of Field Robotics*, vol. 23, pp. 839–861, 2006.

[8] H. Hu, D. Munoz, J. Bagnell, and M. Hebert, "Efficient 3-D scene analysis from streaming data," in *Proc. IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 2297–2304.

[9] V. Plaza-Leiva, J. A. Gomez-Ruiz, A. Mandow, and A. García-Cerezo, "Voxel-based neighborhood for spatial shape pattern classification of lidar point clouds with supervised learning," *Sensors*, vol. 17, no. 594, pp. 1–17, 2017.

[10] Matlab, "Statistics and machine learning toolbox," Available from: https://mathworks.com/products/statistics.html, 2017.

[11] J. Farooq, "Object detection and identification using surf and bow model," in *International Conference on Computing, Electronic and Electrical Engineering*, 2016, pp. 318–323.

[12] M. Maleki, N. Manshouri, and T. Kayikcioglu, "Application of PLSR with a comparison of MATLAB Classification Learner App in using BCI," in *In Proceedings of the 25th Signal Processing and Communications Applications Conference, Antalya, Turkey*, 2017, pp. 1–4.

[13] J. Behley, V. Steinhage, and A. Cremers, "Performance of histogram descriptors for the classification of 3D laser range data in urban environments," in *Proc. IEEE International Conference on Robotics and Automation, Saint Paul (USA)*, Saint Paul, USA, 2012, pp. 4391–4398.

[14] D. Munoz, N. Vandapel, and M. Hebert, "Classification of 3-D point clouds with learned high-order markov random fields," in *Proc. IEEE International Conference on Robotics and Automation, Kobe (Japan)*, Kobe, Japan, 2009.

[15] J.-F. Lalonde, N. Vandapel, and M. Hebert, "Data structures for efficient dynamic processing in 3-D," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 777–796, 2007.

[16] M. Weinmann, S. Urban, S. Hinz, B. Jutzi, and C. Mallet, "Distinctive 2D and 3D features for automated large-scale scene analysis in urban areas," *Computers & Graphics*, vol. 49, pp. 47–57, 2015.

[17] M. Kragh, R. Jorgensen, and H. Pedersen, "Object detection and terrain classification in agricultural fields using 3D lidar data," *Lecture Notes in Computer Science*, vol. 9163, pp. 188–197, 2015.

[18] A. Santamaria-Navarro, E. Teniente, M. Morta, and J. Andrade-Cetto, "Terrain classification in complex three-dimensional outdoor environments," *Journal of Field Robotics*, vol. 32, pp. 42–60, 2015.

[19] A. Birk, K. Pathak, N. Vaskevicius, M. Pfingsthorn, J. Poppinga, and S. Schwertfeger, "Surface representations for 3D mapping," *Kunstliche Intelligenz*, vol. 24, pp. 249–254, 2010.

[20] I. Stamos and M. Leordeanu, "Automated feature-based range registration of urban scenes of large scale," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Madison, USA, 2003, pp. 555–561.

[21] C. Tong, T. Barfoot, and E. Dupuis, "Three-dimensional SLAM for mapping planetary work site environments," *Journal of Field Robotics*, vol. 29, pp. 381–412, 2012.

[22] X. Xiong, D. Munoz, J. Bagnell, and M. Hebert, "3-D scene analysis via sequenced predictions over points and regions," in *Proc. IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2609–2616.

[23] J. Morales, J. L. Martínez, A. Mandow, A. Pequeño-Boter, and A. García-Cerezo, "Design and development of a fast and precise low-cost 3D laser rangefinder," in *Proc. IEEE International Conference on Mechatronics*, Istambul, Turkey, 2011, pp. 621–626.